Refine Search

Search Results -

Terms	Documents
(finite state machine) with ((write adj2 twice) near3 (mode or state))	0

Database: US Pre-Grant Publication Full-Text Database; US Patents Full-Text Database; US OCR Full-Text Database; EPO Abstracts Database; JPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L22

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Saturday, March 05, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
L22	(finite state machine) with ((write adj2 twice) near3 (mode or state))	0	L22
	DB=PGPB,USPT; PLUR=YES; OP=ADJ		
L21	(finite state machine) with ((write adj2 twice) near3 (mode or state))	0	L21
L20	(finite state machine) with ((write adj2 once) near3 (mode or state))	0	L20
L19	(finite state machine) with ((wait or delay$3 or block$3) near3 (mode or state))	198	L19
L18	(finite state machine) with ((idle or inactive) near3 (mode or state))	47	L18
L17	L16 and (control$4 with (finite state machine))	59	L17
L16	(buffer or FIFO) and (table with (finite state machine))	180	L16
L15	(finite state machine) with (advantage or benefit)	41	L15
L14	(buffer or FIFO) with control$4 with (finite state machine) with (advantage or benefit)	0	L14
L13	control$4 with (finite state machine)	2503	L13
L12	l9 and (finite state machine)	2	L12
L11	l9 and ((control$4 near6 (inactive or idle)) with (write near6 (transfer$4 or send$3 or forward$3 or execut$3 or process$3)))	0	L11
L10	l9 and (control$4 near6 (inactive or idle))	37	L10
L9	L8 and (read near6 (delay$3 or stall$4))	162	L9

L8	L5 and (address near6 (compar$6 or match$3 or "same" or correspond$3 or conflict$3))	510	L8
L7	L6 and (read near6 (delay$3 or stall$4))	162	L7
L6	L5 and (address near6 (compar$6 or match$3 or "same" or correspond$3))	510	L6
L5	L4 and ((buffer or FIFO) near6 control$4)	855	L5
L4	L3 and (data near4 bus)	945	L4
L3	L2 and (controller near6 (memory or storage or disk or disc))	1254	L3
L2	(write near3 (buffer or FIFO)) with (transfer$4 or send$3 or forward$3) with (memory or storage or disk or disc) with (read or write or command or request)	1930	L2
L1	(buffer or FIFO) near6 (transfer$4 or send$3 or forward$3) near4 (write near3 data) near8 (memory or storage or disk or disc) with ((accord$4 or based) near6 (read or write or request or command))	3	L1

END OF SEARCH HISTORY

Welcome to IEEE Xplore®

- Home
- What Can I Access?
- Log-out

Tables of Contents

- Journals & Magazines
- Conference Proceedings
- Standards

Search

- By Author
- Basic
- Advanced
- CrossRef

Member Services

- Join IEEE
- Establish IEEE Web Account
- Access the IEEE Member Digital Library

IEEE Enterprise

- Access the IEEE Enterprise File Cabinet

Your search matched **1** of **1131693** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or enteri new one in the text box.

((finite state machine) <near/4> control$) and ((finite Search

☐ Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

1 **A TRD trigger for the Tevatron collider experiment at D0**
Utes, M.; Johnson, M.; Martin, M.;
Nuclear Science Symposium and Medical Imaging Conference, 1991., Confere
Record of the 1991 IEEE , 2-9 Nov. 1991
Pages:589 - 593 vol.1

[Abstract] [PDF Full-Text (424 KB)] **IEEE CNF**

Terms used
finite state machine near/4 control and **finite state machine near/8 lookup table** or **look up table**

Found **43,838** of **151,219**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Efficient instruction scheduling using finite state automata

Vasanth Bala, Norman Rubin

December 1995 **Proceedings of the 28th annual international symposium on Microarchitecture**

Full text available: pdf(1.34 MB) Additional Information: full citation, references, citings, index terms

2 A finite machine description of a lexical analysis using table look-up

D. Soda, G. W. Zobrist

February 1989 **Proceedings of the seventeenth annual ACM conference on Computer science : Computing trends in the 1990's: Computing trends in the 1990's**

Full text available: pdf(193.71 KB) Additional Information: full citation, abstract, references, index terms

A lexical analysis is described using a finite state machine to express the top level of the algorithm. The lexical analyzer exploits table lookup to facilitate the determination of tokens. The table lookup procedure alleviates the process of developing a conplicated set of construction rules.

3 Deterministic part-of-speech tagging with finite-state transducers

Emmanuel Roche, Yves Schabes

June 1995 **Computational Linguistics**, Volume 21 Issue 2

Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings

Publisher Site

Stochastic approaches to natural language processing have often been preferred to rule-based approaches because of their robustness and their automatic training capabilities. This was the case for part-of-speech tagging until Brill showed how state-of-the-art part-of-speech tagging can be achieved with a rule-based tagger by inferring rules from a training corpus. However, current implementations of the rule-based tagger run more slowly than previous approaches. In this paper, we present a finit ...

4 Forth: Forth report: jump tables and state machines

Julian V. Noble

February 2002 **ACM SIGPLAN Notices**, Volume 37 Issue 2

Full text available: pdf(288.50 KB) Additional Information: full citation, references, citings

5 Connection Machine Lisp: fine-grained parallel symbolic processing

Guy L. Steele, W. Daniel Hillis

August 1986

Proceedings of the 1986 ACM conference on LISP and functional programming

Full text available: pdf(1.81 MB) Additional Information: full citation, references, citings

6 Speculative execution and branch prediction on parallel machines

Kevin B. Theobald, Guang R. Gao, Laurie J. Hendren

August 1993 **Proceedings of the 7th international conference on Supercomputing**

Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms

Several recent studies on the limits of parallelism have reported that speculative execution can substantially increase the amount of exploitable parallelism in programs, especially non-numerical programs. This is true even for parallel machines models which allow multiple flows of control. However, most architectural techniques for speculation and branch prediction are geared toward conventional computers with a single flow of control, and little has been done in studying ...

7 Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science

William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J. Schweppe, William Viavant, David M. Young

March 1968 **Communications of the ACM**, Volume 11 Issue 3

Full text available: pdf(6.63 MB) Additional Information: full citation, references, citings

Keywords: computer science academic programs, computer science bibliographies, computer science courses, computer science curriculum, computer science education, computer science graduate programs, computer science undergraduate programs

8 Bootstrapping morphological analyzers by combining human elicitation and machine learning

Kemal Oflazer, Sergei Nirenburg, Marjorie McShane

March 2001 **Computational Linguistics**, Volume 27 Issue 1

Full text available: pdf(1.76 MB) Publisher Site Additional Information: full citation, abstract, references

This paper presents a semiautomatic technique for developing broad-coverage finite-state mor-phological analyzers for use in natural language processing applications. It consists of three components---elicitation of linguistic information from humans, a machine learning bootstrapping scheme, and a testing environment. The three components are applied iteratively until a threshold of output quality is attained. The initial application of this technique is for the morphology of low-density languag ...

9 Automated design of finite state machine predictors for customized processors

Timothy Sherwood, Brad Calder

May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture**, Volume 29 Issue 2

Full text available: pdf(914.12 KB) Additional Information: full citation, abstract, references, citings, index terms

Customized processors use compiler analysis and design automation techniques to take a generalized architectural model and create a specific instance of it which is optimized to a given application or set of applications. These processors offer the promise of satisfying the high performance needs of the embedded community while simultaneously shrinking design times.

Finite State Machines (FSM) are a fundamental building block in computer architecture, and are used to control ...

10 A simple technique for structured variable lookup

Geoffrey W. Gates, David A. Poplawski

September 1973 **Communications of the ACM**, Volume 16 Issue 9

Full text available: pdf(441.86 KB) Additional Information: full citation, abstract, references, citings

A simple technique for the symbol-table lookup of structured variables based on simple automata theory is presented. The technique offers a deterministic solution to a problem which is currently handled in a nondeterministic manner in PL/I and COBOL compilers.

Keywords: PL/I and COBOL structured variables, symbol table organization

11 Translator writing systems

Jerome Feldman, David Gries

February 1968 **Communications of the ACM**, Volume 11 Issue 2

Full text available: pdf(4.47 MB) Additional Information: full citation, abstract, references, citings

A critical review of recent efforts to automate the writing of translators of programming languages is presented. The formal study of syntax and its application to translator writing are discussed in Section II. Various approaches to automating the postsyntactic (semantic) aspects of translator writing are discussed in Section III, and several related topics in Section IV.

Keywords: compiler compiler-compiler, generator, macroprocessor, meta-assembler, metacompiler, parser, semantics, syntactic analysis, syntax, syntax-directed, translator, translator writing system

12 Spoken dialogue technology: enabling the conversational user interface

Michael F. McTear

March 2002 **ACM Computing Surveys (CSUR)**, Volume 34 Issue 1

Full text available: pdf(987.69 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Spoken dialogue systems allow users to interact with computer-based applications such as databases and expert systems by using natural spoken language. The origins of spoken dialogue systems can be traced back to Artificial Intelligence research in the 1950s concerned with developing conversational interfaces. However, it is only within the last decade or so, with major advances in speech technology, that large-scale working systems have been developed and, in some cases, introduced into commerc ...

Keywords: Dialogue management, human computer interaction, language generation, language understanding, speech recognition, speech synthesis

13 Quantum complexity theory

Ethan Bernstein, Umesh Vazirani

June 1993 **Proceedings of the twenty-fifth annual ACM symposium on Theory of computing**

Full text available: pdf(1.13 MB) Additional Information: full citation, references, citings, index terms

14 Query evaluation techniques for large databases

Goetz Graefe

June 1993 **ACM Computing Surveys (CSUR)**, Volume 25 Issue 2

Full text available: pdf(9.37 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Database management systems will continue to manage large data volumes. Thus, efficient algorithms for accessing and manipulating large sets and sequences will be required to provide acceptable performance. The advent of object-oriented and extensible database

systems will not solve this problem. On the contrary, modern data models exacerbate the problem: In order to manipulate large sets of complex objects as efficiently as today's database systems manipulate simple records, query-processi ...

Keywords: complex query evaluation plans, dynamic query evaluation plans, extensible database systems, iterators, object-oriented database systems, operator model of parallelization, parallel algorithms, relational database systems, set-matching algorithms, sort-hash duality

15 Technique for automatically correcting words in text

Karen Kukich

December 1992 **ACM Computing Surveys (CSUR)**, Volume 24 Issue 4

Full text available: pdf(6.23 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Research aimed at correcting words in text has focused on three progressively more difficult problems:(1) nonword error detection; (2) isolated-word error correction; and (3) context-dependent work correction. In response to the first problem, efficient pattern-matching and n-gram analysis techniques have been developed for detecting strings that do not appear in a given word list. In response to the second problem, a variety of general and application-specific spelling cor ...

Keywords: n-gram analysis, Optical Character Recognition (OCR), context-dependent spelling correction, grammar checking, natural-language-processing models, neural net classifiers, spell checking, spelling error detection, spelling error patterns, statistical-language models, word recognition and correction

16 SAFKASI: a security mechanism for language-based systems

Dan S. Wallach, Andrew W. Appel, Edward W. Felten

October 2000 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 9 Issue 4

Full text available: pdf(234.89 KB) Additional Information: full citation, abstract, references, citings, index terms

In order to run untrusted code in the same process as trusted code, there must be a mechanism to allow dangerous calls to determine if their caller is authorized to exercise the privilege of using the dangerous routine. Java systems have adopted a technique called stack inspection to address this concern. But its original definition, in terms of searching stack frames, had an unclear relationship to the actual achievement of security, overconstrained the implementation of a Java system, lim ...

Keywords: Internet, Java, WWW, access control, applets, security-passing style, stack inspection

17 Design of a LISP-based microprocessor

Guy Lewis Steele, Gerald Jay Sussman

November 1980 **Communications of the ACM**, Volume 23 Issue 11

Full text available: pdf(1.89 MB) Additional Information: full citation, abstract, references, citings

We present a design for a class of computers whose "instruction sets" are based on LISP. LISP, like traditional stored-program machine languages and unlike most high-level languages, conceptually stores programs and data in the same way and explicitly allows programs to be manipulated as data, and so is a suitable basis for a stored-program computer architecture. LISP differs from traditional machine languages in that the program/data storage is conceptually an unordered set of ...

Keywords: LISP, SCHEME, VLSI, direct execution, garbage collection, high-level language architectures, integrated circuits, interpreters, large-scale integration, linked lists, list structure, microprocessors, storage management, tail recursion

18 A parallel/serial trade-off methodology for look-up table based decoders

Claus Schneider

June 1997 **Proceedings of the 34th annual conference on Design automation - Volume 00**

Full text available: pdf(60.71 KB) Additional Information: full citation, abstract, references, index terms

Publisher Site

A methodology for architecture exploration of look-up tablebased decoders is presented. For the degree of parallel processinga trade-off can be made by exploring system leveland register transfer level models. Executable specifications(pure functional software models, VHDL behavior models)are used to analyze the performance of different architectures.Hardware cost (area) and feasibility (timing) aredetermined by synthesis of RTL models. These models aregenerated directly out of the specification ...

19 Parallel execution of prolog programs: a survey

Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, Manuel V. Hermenegildo

July 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 4

Full text available: pdf(1.95 MB) Additional Information: full citation, abstract, references, citings, index terms

Since the early days of logic programming, researchers in the field realized the potential for exploitation of parallelism present in the execution of logic programs. Their high-level nature, the presence of nondeterminism, and their referential transparency, among other characteristics, make logic programs interesting candidates for obtaining speedups through parallel execution. At the same time, the fact that the typical applications of logic programming frequently involve irregular computatio ...

Keywords: Automatic parallelization, constraint programming, logic programming, parallelism, prolog

20 A Four Russians algorithm for regular expression pattern matching

Gene Myers

April 1992 **Journal of the ACM (JACM)**, Volume 39 Issue 2

Full text available: pdf(1.30 MB) Additional Information: full citation, abstract, references, citings, index terms

Given a regular expression R of length P and a word A of length N, the membership problem is to determine if A is in the language denoted by R. An O(PN/lgN) time algorithm is presented that is based on a lgN speedup of the standard O(PN) time simulation of R's nonderministi ...

Keywords: Four Russians paradigm, finite automaton, node listing, regular expression